Extracting Content Bearing Terms in Parallel on the Connection Machine®

S. Smith

Thinking Machines Corporation

Technical Report Series

TMC-71

Extracting Content Bearing Terms in Parallel on the Connection Machine

Stephen Smith
Thinking Machines Corporation
245 First Street
Cambridge MA 02142
(617) 876-1111

ABSTRACT

The value of the Connection Machine System [1] for fast and precise retrieval of relevant documents has already been demonstrated [2]. Sixteen thousand newspaper articles can now be searched for 200 weighted search terms in about 60 milliseconds. This paper will discuss algorithms for processing text into forms suitable for document search; specifically in developing fast methods for extracting content-bearing terms from raw text via frequency information. There are three main topics: 1. Creating a frequency dictionary from the raw text. 2. Using the dictionary to classify terms in the raw text. 3. Marking word boundaries and identifying proper nouns.

Paper-type: Full-Paper
Topic area: Perception and Signal Understanding - data interpretation
Track: Engineering
Keywords: Data Compression, Parallel Computing, Information Retrieval, Connection Machine System

Extracting Content Bearing Terms in Parallel on the Connection Machine

1 The Connection Machine System

The Connection Machine is a fine-grained parallel computer with a maximum configuration of 65,536 processing elements. Each processing element contains 4,096 bits of memory and a 1 bit wide ALU. All elements are connected via a 12 dimensional hypercube network which contains 16 processing elements at each vertex. Any processor can communicate with any other processor via this network.

A user can interface to the Connection Machine via a front-end machine; either a Symbolics 3600 Series Lisp Machine or a Digital Equipment VAX. The original environments of the front end machines are preserved so that a user can develop code with known tools and within a familiar operating system. The programming languages for the Connection Machine (currently an enhanced version of C and Common Lisp) are extended to include functions appropriate for the Connection Machine.

An important parallel function that is referenced throughout this paper is *Scan*[3]. Scan makes use of the hypercube network to allow various functions to be performed on many processors in logarithmic time. Thus a field within each processor could be summed across all 65,536 processors in just 16 applications of *plus*.

Currently user callable functions in the scan family include PLUS-SCAN (for adding values in many processors), COPY-SCAN (for copying a value from one processor to many others), and MAX-SCAN (for finding the largest value in any of a large number of processors). Additionally these functions allow for the specification of a segment bit that delimits the start position of groups of processors across the machine. These groups can then have the scan operations applied simultaneously (eg. sum within each of a number of groups of processors at the same time). Segmented scans also require time on the order of log n.

2 The Seed Document Retrieval System

The Seed Document Retrieval System currently running on a 16k Connection Machine rapidly searches a database of 3 months of news articles from the Reuters newswire (approximately 16,000 articles = 32 Mbytes) [2]. To perform a search of the database for a particular news topic a user employs two different methods. To initiate the search a user inputs a number of key words he believes should be contained in documents relevant to

the current topic. Documents containing these words are then located in the database and displayed in ranked order from most to least relevant. At this point the user can employ a second method of search called "Relevance Feedback" [4].

By marking one or more of the documents returned from the initial query as either GOOD or BAD, a user can further direct his search by informing the system as to which documents are and are not relevant. The retrieval system then uses all the terms from the marked documents as seeds to continue and refine the search.

The structure of the data on the Connection Machine can be simply visualized as one newspaper article residing in each processor. The search proceeds by broadcasting each word from the user's query to all processors in parallel. Each processor then looks at the document that it contains and determines whether the broadcast word is contained in the document. If the word is in the document then the score of the document is increased; this document score is used to determine the ranked ordering of documents once all the terms have been broadcast.

The ultimate score of a document is determined by the weights of the words that it contains. Each word being assigned a different weight value depending in large part on the frequency with which the word appears in the database. For example assume a user starts off his query with the terms "MARCOS", "HIDDEN" and "WEALTH" in looking for articles concerning the secret assets of Ferdinand Marcos. Each of these words is assigned a fairly high weight in that each is part of the initial query. "MARCOS" would be assigned the highest weight since it does not occur very often in the database and thus is very helpful in distinguishing between documents that do and do not pertain to this topic.

In the relevance feedback phase of the search all content bearing words in marked documents are broadcast to the Connection Machine. This means that in an average search (after 2 or 3 documents are marked good or bad) 200 or more words are broadcast to the system - resulting in a much better search of the database. The weight of each word in this phase is further influenced by whether it appears in a GOOD or BAD document; words appearing in GOOD documents having a positive weight and those appearing only in BAD documents having a negative weight.

3 Introduction To The Problem

In order for the Document Retrieval system to work as well as it does it is necessary for the initial raw text to be processed through a number of conditioning steps before it is placed on the Connection Machine. The current size of our Reuters database is 140 Mbytes, thus it is important that all pre-processing be performed as efficiently as possible.

The initial text received from the Reuters newswire is in a human readable ASCII format. While this raw text form is fine for people reading the newspaper, it is an inappropriate format for performing efficient searches on the Connection Machine. The raw text form is also a space-inefficient way to store the data, in that a minority of the words in the original text contain a majority of the information. Specifically words like "the" and "a", prepositions (above, in, to, on), adverbs (always, very, never, however, moreover), and auxilliaries (is, be, were, must, should) can be eliminated from the text with almost no loss in performance. Thus compressing the initial data without loss of information is an important task. The real problem then is in determining which words to save and which words to throw out.

4 Data Compression Techniques

Initially data compression for the Document Retrieval System was performed serially on a Symbolics 3600 computer. The algorithms used were those developed for the ConText Text Indexer and were meant more specifically for producing a human readable back-of-the-book style index than for determining which of many individual words were and were not content bearing. The compression algorithms described here are parallel and use very few heuristics or "understanding" of sentence structure; instead relatively good results have been obtained based primarily on word frequency information. These algorithms have the further advantage that they are simple and easily implemented on a parallel machine.

To get an idea of how well frequency based indexing works look at a portion of a news article from Reuters below. The words that are underlined are among the 30% least frequent words and those that are in italics are among the 40% most common words.

ROME COURT OFFICIALS SET *TO* QUESTION BULGARIANS *ON* POPE PLOT
By Andrew Hurst

SOFIA, Bulgaria, Dec 20, *Reuter* - *An* Italian judge *and* prosecutor *were* due *today to* question *two* Bulgarians accused *of* participating *in a* conspiracy *to* kill Pope John Paul II. *The* accused, diplomats Todor Aivazov *and* Zhelyo Vassilev, *are* among three Bulgarians *and* three Turks *on* trial *in* Italy charged *with* conspiring *with* Turkish gunman Mehmet Ali Agca *to* kill *the* pope. Aivazov *and* Vassilev *are* being tried *in their* absence *after* Bulgaria refused *to* extradite them *on the* grounds *of* diplomatic immunity. *They* left Italy *in* 1982, *more* than *a year after* Agca shot *and* wounded *the* pope *in* St. Peters Square *on* May 13, 1981.

By throwing out the top 40% most frequent words we can be fairly well assured that no content bearing terms will be lost. It also appears that almost no non-content-bearing terms are included in the 30% least common words. Words that fall between the top 30% and bottom 40% cutoffs can be subselected by noting whether they appear multiple

times in a given article (in the example article this can be seen in the word "pope" which falls between the two cutoffs but is marked because it appears many times in the article). When proper nouns are also marked we can be assured of having a high percentage of all content-bearing terms and only a few non-content-bearing terms:

ROME COURT OFFICIALS SET *TO* QUESTION BULGARIANS *ON* POPE PLOT
By Andrew Hurst

SOFIA, Bulgaria, Dec 20, *Reuter - An* Italian judge *and* prosecutor *were* due *today to* question *two* Bulgarians accused *of* participating *in a* conspiracy *to* kill Pope John Paul II. *The* accused, diplomats Todor Aivazov *and* Zhelyo Vassilev, *are* among three Bulgarians *and* three Turks *on* trial *in* Italy charged *with* conspiring *with* Turkish gunman Mehmet Ali Agca *to* kill *the* pope. Aivazov *and* Vassilev *are* being tried *in their* absence *after* Bulgaria refused *to* extradite them *on the* grounds *of* diplomatic immunity. *They* left Italy *in* 1982, *more* than *a year after* Agca shot *and* wounded *the* pope *in* St. Peters Square *on* May 13, 1981.

These techniques can be enhanced by creating an explicit drop list (words like "the" and "to" are almost never helpful), or creating a thesaurus of words that commonly occur together - (thus "kill", "shot" and "wounded" could be noted as related and could thus be retained in the example article). Morphological information can also be exploited to note, for example, that "diplomats" and "diplomatic" are similar and that if one is considered content bearing, then the other should be also.

5 Algorithms For The Connection Machine

To select content bearing terms via frequency it is first necessary to build a frequency dictionary (a dictionary whose definitions contain the frequency of the given word in the database). The dictionary is then used to assign frequency values to each word of the text. In the future this dictionary will also contain morphological information and a thesaurus.

An overview of the content-bearing term extraction process is as follows:

I. Build the dictionary by processing all raw text:
   1. Load the text to be processed onto the Connection Machine, with one character per processor.
   2. Note word breaks.
   3. Accumulate all the characters of a word into a single processor.
   4. Move these words into the dictionary.
   5. Sort the dictionary alphabetically and then sum words that are identical.

II. Use the dictionary to mark content bearing words:
   1. Reload the text with one character per processor.
   2. Note word breaks and proper nouns.

3. Accumulate all the characters of a word into a single processor.
4. Move these words into the dictionary keeping a pointer to their original location.
5. Copy-scan definitions to the new words.
6. Send the definitions back to the words at their initial positions.
7. Mark the content bearing terms by comparing their frequencies to the upper and lower frequency cutoffs.

6 Building The Dictionary

A dictionary of words and information must be available on the CM in order to accurately select terms via frequency. The dictionary described here has only two entries - the word itself and the number of times the word appears in the given data base. A finished dictionary looks like this with one word and its frequency stored in each processor:

proc-#	0	1	2	...	1500	...	65535
word	the	a	tried	...	judge	...	bulgarians
frequency	10000	5000	1000	...	100	...	5

The initial text on the Connection Machine looks like this:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	...
T	h	e		j	u	d	g	e		t	r	i	e	d		t	h	e		B	u	l	g	...

To determine beginings and endings of words in the text a simple heuristic is used: *If the character is a letter then it is considered to be part of a word. If a character is a hyphen then it is part of a word only if the characters to either side of it are letters.* Each processor can check its own contents in parallel and it can check the contents of its neighbors by using the hypercube communications network. The processor notes itself as part of a word by setting one of its bits, the WORD-BIT, to 1. This bit can later be used for determining the length of the word.

	0	1	2	3	4	5	6	7	8	9	
word-bit	1	1	1	0	1	1	1	1	1	0	...
text-char	T	h	e		j	u	d	g	e		...
left-nghbr.		T	h	e		j	u	d	g	e	...
right-nghbr.	h	e		j	u	d	g	e		t	...

The technique for marking proper nouns is only slightly more elaborate than that of marking word boundaries. Proper nouns are defined as *those words that are capitalized and are not preceded by either a period and a carriage return or a period and a space* (this heuristic will, of course, be specific to the text that is currently being processed). Thus by having each processor look at itself and its two preceding neighbors it can accurately set or clear a PROPER-NOUN bit.

To perform an alphabetic sort on the Connection Machine it is most efficient to sort with one word per processor, thus the characters presently distributed across processors must be accumulated into one processor. One way to do this is to perform a send for each character in a word using the hypercube communications network . This method, however, requires doing as many sends as there are characters in each word. Send is, however, a relatively expensive operation and should be limited as much as possible. Another solution, that requires only, log n sends is shown below. An understanding of this technique should also shed light on how scan operates in log time.

Stack neighbors:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	...
T	h	e		**j**	u	d	g	e		**t**	r	i	e	d		**t**	h	e		**B**	u	l	g	...
h	e		j	**u**	d	g	e		t	**r**	i	e	d		t	**h**	e		B	**u**	l	g	a	...

Stack neighbors 2 away:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	...
T	h	e		**j**	u	d	g	e		**t**	r	i	e	d		**t**	h	e		**B**	u	l	g	...
h	e		j	**u**	d	g	e		t	**r**	i	e	d		t	**h**	e		B	**u**	l	g	a	...
e		j	u	**d**	g	e		t	r	**i**	e	d		t	h	**e**		B	u	**l**	g	a	r	...
	j	u	d	**g**	e		t	r	i	**e**	d		t	h	e		B	u	l	**g**	a	r	i	...

Continuing the stack in this way (1 2 4 8 ..) the final accumulated words would look like this:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	...
T	h	e		j	u	d	g	e		t	r	i	e	d		t	h	e		B	u	l	g	...
The				judge						tried						the				Bulgarians				...

After these words have been set up they must be moved into the dictionary where they are sorted alphabetically:

proc-#	0	1	2	3	4	5	6	7	8
word	a	bulgarians	judge	judge	the	the	the	tried	tried
frequency	5000	1	500	1	10000	1	1	1000	1
segment-bit	1	1	1	0	1	0	0	1	0

Plus-Scan

Using the segmented plus-scan function sum up the number of times a given word appears in the dictionary:

proc-#	0	1	2	3	4	5	6	7	8
word	a	Bulgarians	judge	judge	the	the	the	tried	tried
frequency	5000	1	500	501	10000	10001	10002	1000	1001

7 Using The Dictionary

Techniques used in propagating frequency values from dictionary entries to new words are similar to those used in creating the dictionary [5]. All characters of each word must be first accumulated into a single processor and this word then inserted into the completed dictionary. In this case, however, a pointer to the word's original location is moved with the word.

Let us start from the point where the new words have already been inserted into the dictionary. Note the dictionary entries do not have any value in their HOME field (original address of the new word) and the new words to be defined have no value in their FREQUENCY field. Again we perform an alphabetic sort but this time a one bit key field is added to each word and its value is set to 1 in non-dictionary entries. Now when the sort is performed it is assured that the dictionary entry is the first in any series of identical words. It is important to have the dictionary entry precede the new word entries in that to propagate the dictionary definitions to the new words a segmented copy-scan is used (the segment start being set wherever the preceding word is different).

proc-#	0	1	2	3	4	5	6	7	8
word	a	bulgarians	judge	judge	the	the	the	tried	tried
frequency	7890	-	675	675	22003	22003	22003	1452	1452
home	-	20	-	4	-	0	16	-	10
sort-key	a:0	bulgarians:1	judge:0	judge:1	the:0	the:1	the:1	tried:0	tried:1
segment-bit	1	1	1	0	1	0	0	1	0

Copy-Scan

Finally the words with their values must be sent back to the processors that they originated from via the home address that they have retained. Our original text should now look something like this:

proc-#	0	4	10	16	20
word	The	judge	tried	the	Bulgarians
frequency	22003	675	1452	22003	-

Now apply the upper and lower frequency thresholds marking a KEEP and DUMP bits accordingly. To mark multiple occurrences of a given word that falls between the two thresholds a segmented alphabetic sort is performed. Thus identical words within each document are placed side by side. To determine multiple occurrences within a document each processor looks to see if its neighbors are identical. If they are and the word contained in the processor does not have its DUMP bit set then its KEEP bit is set.

8 Conclusion

Extraction of content bearing terms from text can thus be performed efficiently on the Connection Machine System. Further research on this extraction problem will most likely center around the use of the current data structure of the frequency dictionary and its related retrieval algorithms. Incorporating a thesaurus and morphological information will hopefully produce a system capable of performing more challenging tasks in natural langauage processing in the near future.

References

1. D. Hillis, *The Connection Machine*, MIT Press, Cambridge, Mass., 1985.
2. C. Stanfill and B. Kahle, "Parallel Free Text Search on the Connection Machine System," *Comm. ACM*, Vol. 29 No. 12, Dec. 1986
3. D. Hillis and G. L. Steele, Jr., "Data Parallel Algorithms," *Comm. ACM*, Vol. 29, No. 12, Dec. 1986.
4. Salton, G. *The SMART Retrieval System - Experiment in Automatic Document Processing.* Prentice-Hall, Englewood Clifs, N.J., 1971
5. G. Sabot, "Bulk Processing of Text on a Massively Parallel Computer," *Proc. 24th Annual Meeting of the Association for Computer Linguistics*, New York, June 10-13, 1986, pp. 128-135